Last Time

- Two multicast algorithms for total ordering
  - Sequencer
  - ISIS
- Multicast for causal ordering
  - Uses vector timestamps

Review: Causally Ordered Multicast

- Each process keeps a vector clock.
  - Each counter represents the number of messages received from each of the other processes.
- When multicasting a message, the sender process increments its own counter and attaches its vector clock.
- Upon receiving a multicast message, the receiver process waits until it can preserve causal ordering:
  - It has delivered all the messages from the sender.
  - It has delivered all the messages that the sender had delivered before the multicast message.

Review: Causal Ordering

Algorithm for group member $p_i$ ($i = 1, 2, \ldots, N$)

On initialization $V_i^0[j] = 0$ ($j = 1, 2, \ldots, N$).

To CO-multicast message $m$ to group $g$

$V_i^0[i] = V_i^0[i] + 1$;

$B$-multicast($g, < V_i^0, m>$);

On $B$-deliver($< V_i^0, m>$) from $p_j$, with $g = \text{group}(m)$

place $< V_j^0, m>$ in hold-back queue;

wait until $V_i^0[j] = V_j^0[j] + 1$ and $V_i^0[k] \leq V_j^0[k]$ ($k \neq j$);

CO-deliver $m$ // after removing it from the hold-back queue

$V_i^0[j] = V_i^0[j] + 1$;

Example: Causal Ordering Multicast

Today’s Question

- How do we organize the nodes in a distributed system?
- Up to the 90’s
  - Prevalent architecture: client-server (or master-slave)
  - Unequal responsibilities
- Now
  - Emerged architecture: peer-to-peer
  - Equal responsibilities
- Studying an example client-server: DNS (today)
- Studying peer-to-peer as a paradigm (not just as a file-sharing application)
  - Learn the techniques and principles
Separating Names and IP Addresses

- Names are easier (for us!) to remember
  - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

Two Kinds of Identifiers

- Host name (e.g., www.cnn.com)
  - Mnemonic name appreciated by humans
  - Provides little (if any) information about location
  - Hierarchical, variable # of alpha-numeric characters
- IP address (e.g., 64.236.16.20)
  - Numerical address appreciated by routers
  - Related to host’s current location in the topology
  - Hierarchical name space of 32 bits

Hierarchical Assignment Processes

- Host name: www.cse.buffalo.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- IP addresses: 128.205.32.58
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP

Domain Name System (DNS)

Proposed in 1983 by Paul Mockapetris

Overview: Domain Name System

- A client-server architecture
  - The server-side is still distributed for scalability.
  - But the servers are still a hierarchy of clients and servers
- Computer science concepts underlying DNS
  - Indirection: names in place of addresses
  - Hierarchy: in names, addresses, and servers
  - Caching: of mappings from names to/from addresses
- DNS software components
  - DNS resolvers
  - DNS servers
- DNS queries
  - Iterative queries
  - Recursive queries
- DNS caching based on time-to-live (TTL)

Strawman Solution #1: Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: moving from a machine per domain to machine per user
  - Many more downloads
  - Many more updates
Strawman Solution #2: Central Server

• Central server
  – One place where all mappings are stored
  – All queries go to the central server
• Many practical problems
  – Single point of failure
  – High traffic volume
  – Distant centralized database
  – Single point of update
  – Does not scale

Need a distributed, hierarchical collection of servers

Domain Name System (DNS)

• Properties of DNS
  – Hierarchical name space divided into zones
  – Distributed over a collection of DNS servers
• Hierarchy of DNS servers
  – Root servers
  – Top-level domain (TLD) servers
  – Authoritative DNS servers
• Performing the translations
  – Local DNS servers
  – Resolver software

DNS Root Servers

• 13 root servers (see http://www.root-servers.org/)
• Labeled A through M

TLD and Authoritative DNS Servers

• Top-level domain (TLD) servers
  – Generic domains (e.g., com, org, edu)
  – Country domains (e.g., uk, fr, ca, jp)
  – Typically managed professionally
    » Network Solutions maintains servers for "com"
    » Educause maintains servers for "edu"
• Authoritative DNS servers
  – Provide public records for hosts at an organization
  – For the organization’s servers (e.g., Web and mail)
  – Can be maintained locally or by a service provider

Using DNS

• Local DNS server ("default name server")
  – Usually near the end hosts who use it
  – Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
• Client application
  – Extract server name (e.g., from the URL)
  – Do gethostbyname() to trigger resolver code
• Server application
  – Extract client IP address from socket
  – Optional gethostbyaddr() to translate into name
CSE 486/586 Administrivia

• Please start PA2 if you haven’t.
• AWS codes are distributed on UBLearns.
  – Will post setup instructions.
• Practice problem set 1 & midterm example posted on
  the course website.
• Moving the midterm from Friday (3/8) to Wednesday
  (3/6)?
• Come talk to me!

Example

Host at cis.poly.edu
wants IP address for
gaiac.umaass.edu

Recursive vs. Iterative Queries

• Recursive query
  – Ask server to get
    answer for you
  – E.g., request 1 and
    response 8

• Iterative query
  – Ask server who
    to ask next
  – E.g., all other request-
    response pairs

DNS Caching

• Performing all these queries take time
  – And all this before the actual communication takes place
  – E.g., 1-second latency before starting Web download

• Caching can substantially reduce overhead
  – The top-level servers very rarely change
  – Popular sites (e.g., www.cnn.com) visited often
  – Local DNS server often has the information cached

• How DNS caching works
  – DNS servers cache responses to queries
  – Responses include a “time to live” (TTL) field
  – Server deletes the cached entry after TTL expires

Negative Caching

• Remember things that don’t work
  – Misspellings like www.cnn.com and www.cnnm.com
  – These can take a long time to fail the first time
  – Good to remember that they don’t work
  – … so the failure takes less time the next time around

DNS Resource Records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

• Type=A
  – name is hostname
  – value is IP address

• Type=NS
  – name is domain
  (e.g. foo.com)
  – value is hostname of
    authoritative name server for
    this domain

• Type=CNAME
  – name is alias for some
    “canonical” (the real) name:
    www.ibm.com is really
    srveast.backup2.ibm.com
  – value is canonical name

• Type=MX
  – value is name of mailserver
    associated with name
Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

Inserting Resource Records into DNS

- Example: just created startup “FooBar”
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RR’s into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
  - Type MX record for foobar.com
- Play with “dig” on UNIX

$ dig nytimes.com ANY
; QUESTION SECTION:
:nytimes.com. IN ANY

; ANSWER SECTION:
nytimes.com. 247 IN MX 100
 nytimes.com. 247 IN MX 200
 nytimes.com. 247 IN A 199.239.137.200
 nytimes.com. 247 IN A 199.239.138.200
 nytimes.com. 247 IN TXT "v=spf1 mx ptr -ip4:199.239.138.0/24 include:alerts.wallst.com include:authsmtp.com -all"


; AUTHORITY SECTION:

; ADDITIONAL SECTION:

$ dig nytimes.com +norec @a.root-servers.net
; QUESTION SECTION:
:nytimes.com. IN A

; AUTHORITY SECTION:
nytimes.com. 172800 IN NS K.GTLD-SERVERS.NET.
nytimes.com. 172800 IN NS E.GTLD-SERVERS.NET.
nytimes.com. 172800 IN NS C.GTLD-SERVERS.NET.
nytimes.com. 172800 IN NS I.GTLD-SERVERS.NET.
nytimes.com. 172800 IN NS D.GTLD-SERVERS.NET.
nytimes.com. 172800 IN AAAA 2001:503:a83e::2:30

; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET. 172800 IN A 192.5.6.30
A.GTLD-SERVERS.NET. 172800 IN AAAA 2001:503:a83e::2:30
B.GTLD-SERVERS.NET. 172800 IN A 192.33.14.30
B.GTLD-SERVERS.NET. 172800 IN AAAA 2001:503:231d::2:30

$ dig nytimes.com +norec @a.root-servers.net
; QUESTION SECTION:
:nytimes.com. IN ANY

; ANSWER SECTION:
nytimes.com. 300 IN NS ns1t.foobar.com.
nytimes.com. 300 IN NS ns1t.about.com.
nytimes.com. 300 IN NS ns2t.about.com.

; ADDITIONAL SECTION:
ns1t.nytimes.com. 172800 IN A 199.239.137.15
ns2t.nytimes.com. 172800 IN A 199.239.138.15
ns1t.foobar.com. 172800 IN A 199.239.137.245
ns2t.foobar.com. 172800 IN A 199.239.138.245
nytimes.com. 172800 IN TXT "v=spf1 mx ptr -ip4:199.239.138.0/24 include:alerts.wallst.com include:authsmtp.com -all"

$ dig nytimes.com +norec @k.gtld-servers.net
; QUESTION SECTION:
:nytimes.com. IN ANY

; ANSWER SECTION:
Content Distribution Networks (CDNs)

- Content providers are CDN customers
- Content replication
  - CDN company installs thousands of servers throughout Internet
    - In large datacenters
    - Or, close to users
  - CDN replicates customers’ content
  - When provider updates content, CDN updates servers

Server Selection

- Which server?
  - Lowest load: to balance load on servers
  - Best performance: to improve client performance
    - Based on what? Location? RTT? Throughput? Load?
  - Any alive node: to provide fault tolerance
- How to direct clients to a particular server?
  - As part of routing: anycast, cluster load balancer
  - As part of application: HTTP redirect
  - As part of naming: DNS

How Akamai Works

End-user

- DNS lookup
  - cache.cnn.com
  - g.akamai.net

- GET
  - index.html

- HTTP
  - http://cache.cnn.com/
  - cnn.com/fox.jpg

- Akamai cluster
  - Akamai global DNS server
  - Akamai regional DNS server
  - Arbitrary Akamai cluster

HTTP

End-user

- DNS lookup
  - g.akamai.net

- DNS lookup
  - a73.g.akamai.net

- HTTP
  - http://a73.g.akamai.net

- Akamai cluster
  - Akamai global DNS server
  - Akamai regional DNS server
  - Arbitrary Akamai cluster
**Summary**

- DNS as an example client-server architecture
- **Why?**
  - Names are easier (for us!) to remember
  - IP addresses can change underneath
  - Name could map to multiple IP addresses
  - Map to different addresses in different places
  - Multiple names for the same address
- **Properties of DNS**
  - Distributed over a collection of DNS servers
- **Hierarchy of DNS servers**
  - Root servers, top-level domain (TLD) servers, authoritative DNS servers

**Acknowledgements**

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Michael Freedman (Princeton), and Jennifer Rexford (Princeton).